**实验一 进程调度模拟算法**

课程名称 操作系统 信息科学与技术 学院 软件工程 学院 信1405-1班 学号 20143048

实验者姓名 张作秀 实验日期 2017 年 4 月 10 日

评分 教师签名

### 实验一 进程调度模拟算法

一、目的和要求

进程调度是处理机管理的核心内容。本实验要求用高级语言编写模拟进程调度程序，以便加深理解有关进程控制快、进程队列等概念，并体会和了解优先数算法和时间片轮转算法的具体实施办法。

二、实验内容

1.设计进程控制块 PCB 的结构，通常应包括如下信息：

进程名、进程优先数（或轮转时间片数）、进程已占用的 CPU 时间、进程到完成还需要的时间、进程的状态、当前队列指针等。

2.编写两种调度算法程序：

优先数调度算法程序循环轮转调度算法序

3.按要求输出结果。

1. 实验过程

1.准备

1. 查阅相关资料
2. 初步编写程序
3. 准备测试程序

2.上机调试

3.主要流程和源代码

分别用两种调度算法对伍个进程进行调度。每个进程可有三种状态；执行状态（RUN）、就绪状态（READY,包括等待状态）和完成状态（FINISH），并假定初始状态为就绪状态。

**（一）进程控制块结构如下：**

NAME——进程标示符

PRIO/ROUND——进程优先数/进程每次轮转的时间片数（设为常数 2）CPUTIME——进程累计占用 CPU 的时间片数NEEDTIME——进程到完成还需要的时间片数

STATE——进程状态NEXT——链指针

注：

1.为了便于处理，程序中进程的的运行时间以时间片为单位进行计算；

2.各进程的优先数或轮转时间片数，以及进程运行时间片数的初值，均由用户在程序运行时给定。

**（二）进程的就绪态和等待态均为链表结构，共有四个指针如下：**

RUN——当前运行进程指针

READY——就需队列头指针

TAIL—— 就需队列尾指针

FINISH—— 完成队列头指针

**（三）程序说明**

1. 在优先数算法中，进程优先数的初值设为：

50-NEEDTIME

每执行一次，优先数减 1，CPU 时间片数加 1，进程还需要的时间片数减 1。

在轮转法中，采用固定时间片单位（两个时间片为一个单位），进程每轮转一次，CPU时间片数加 2，进程还需要的时间片数减 2，并退出 CPU，排到就绪队列尾，等待下一次调度。

2. 程序的模块结构提示如下：

整个程序可由主程序和如下 7 个过程组成：

（1）INSERT1——在优先数算法中，将尚未完成的 PCB 按优先数顺序插入到就绪队列

中；

（2）INSERT2——在轮转法中，将执行了一个时间片单位（为 2），但尚未完成的进程的 PCB，插到就绪队列的队尾；

（3）FIRSTIN——调度就绪队列的第一个进程投入运行；（4）PRINT——显示每执行一次后所有进程的状态及有关信息。

（5）CREATE——创建新进程，并将它的 PCB 插入就绪队列；（6）PRISCH——按优先数算法调度进程；（7）ROUNDSCH——按时间片轮转法调度进程。

主程序定义 PCB 结构和其他有关变量。

**（四）运行和显示**

程序开始运行后，首先提示：请用户选择算法，输入进程名和相应的 NEEDTIME 值。每次显示结果均为如下 5 个字段：

name cputime needtime priority state

注：

1．在 state 字段中，"R"代表执行态，"W"代表就绪（等待）态，"F"代表完成态。

2．应先显示"R"态的，再显示"W"态的，再显示"F"态的。

3．在"W"态中，以优先数高低或轮转顺序排队；在"F"态中，以完成先后顺序排队。

2.源代码：

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include<windows.h>

typedef struct node

{

char name[10];

int prio;

int round;

int cputime;

int needtime;

int count;

char state;

struct node \*next;

}PCB;

PCB \*finish,\*ready,\*tail,\*run;

int N;

void firstin(void);

void print1(char a);

void print2(char chose,PCB \*p);

void print(char chose);

void insert\_prio(PCB \*q);

void prior\_init(char chose);

void priority(char chose);

void insert\_rr(PCB \*q);

void roundrun\_init(char chose);

void roundrun(char chose);

void main()//主函数

{

char chose=' ';

while((chose!='q')&&(chose!='Q'))

{

fflush(stdin);

printf("选择优先数算法请输入P，选择循环轮转算法请输入R，退出请输入Q\n");

printf("请输入你的选择：");

scanf("%c",&chose);

if((chose!='q')&&(chose!='Q'))

{

system("cls");

if((chose=='P')||(chose=='p'))

{

prior\_init(chose);

priority(chose);

system("cls");

}

else if((chose=='r')||(chose=='R'))

{

roundrun\_init(chose);

roundrun(chose);

system("cls");

}

}

}

printf("谢谢使用！\n");

}

void firstin(void)

{

if(ready!=NULL)

{

run=ready;

ready=ready->next;

run->state='R';

run->next=NULL;

}

else

{

run=NULL;

}

}

void print1(char a)

{

if(toupper(a)=='P')

{

printf("name cputime needtime priority state \n");

}

else

{

printf("name cputime needtime count round state \n");

}

}

void print2(char chose,PCB \*p)

{

if(toupper(chose)=='P')

{

printf("%s\t%d\t%d\t%d\t%c\n",p->name,p->cputime,p->needtime,p->prio,p->state);

}

else

{

printf("%s\t%d\t%d\t%d\t%d\t%c\n",p->name,p->cputime,p->needtime,p->count,p->round,p->state);

}

}

void print(char chose)

{

PCB \*p;

print1(chose);

if(run!=NULL)

{

print2(chose,run);

}

p=ready;

while(p!=NULL)

{

print2(chose,p);

p=p->next;

}

p=finish;

while(p!=NULL)

{

print2(chose,p);

p=p->next;

}

}

void insert\_prio(PCB \*q)

{

PCB \*p,\*s,\*r;

s=q;

p=ready;

r=p;

if(s->prio>ready->prio) {

s->next=ready;

ready=s;

}

else

{

while(p)

{

if(p->prio>=s->prio)

{

r=p;

p=p->next;

}

else

break;

} s->next=p;

r->next=s;

}

}

void prior\_init(char chose)

{

PCB \*p;

int i,time;

char na[10];

ready=NULL;

finish=NULL;

run=NULL;

printf("输入进程 的个数 N:\n");

scanf("%d",&N);

for(i=0;i<N;i++)

{

p=(PCB\*)malloc(sizeof(PCB));

printf("输入第%d个进程名\n",i+1);

scanf("%s",na);

printf("完成进程需要的时间片数\n");

scanf("%d",&time);

strcpy(p->name,na);

p->cputime=0;

p->needtime=time;

p->state='W';

p->prio=50-time;

if(ready==NULL)

{

ready=p;

ready->next=NULL;

}

else

{

insert\_prio(p);

}

printf("当前就绪队列的进程的信息\n");

print(chose);

}

printf("%d个进程已按优先级从高到低进到就绪队列中\n",N);

printf("按回车键开始模拟优先级算法.....\n");

fflush(stdin);

getchar();

firstin();

}

void priority(char chose)

{

int i=1;

while(run!=NULL)

{

run->cputime+=1;

run->needtime-=1;

run->prio-=1;

if(run->needtime==0)

{

run->next=finish;

finish=run;

run->state='F';

run->prio=0;

run=NULL;

firstin();

}

else

{

if((ready!=NULL)&&(run->prio<ready->prio))

{

run->state='W';

insert\_prio(run);

run=NULL;

firstin();

}

}

print(chose);

}

getchar();

}

void insert\_rr(PCB \*q)

{

tail->next=q;

tail=q;

q->next=NULL;

}

void roundrun\_init(char chose)

{

PCB \*p;

int i,time;

char na[10];

ready=NULL;

finish=NULL;

run=NULL;

printf("\t\t循环轮转算法模拟全过程\n\n");

printf("输入进程 的个数 N:\n");

scanf("%d",&N);

for(i=0;i<N;i++)

{

p=(PCB\*)malloc(sizeof(PCB));

printf("输入第%d个进程名\n",i+1);

scanf("%s",na);

printf("完成进程需要的时间片数\n");

scanf("%d",&time);

strcpy(p->name,na);

p->cputime=0;

p->needtime=time;

p->count=0;

p->state='W';

p->round=2;

if(ready!=NULL)

{

insert\_rr(p);

}

else

{

p->next=ready;

ready=p;

tail=p;

}

printf("当前就绪队列的进程的信息\n");

print(chose);

}

printf("%d个进程已按FIFO进到就绪队列中\n",N);

printf("按回车键开始模循环轮转算法.....\n");

fflush(stdin);

getchar();

run=ready;

ready=ready->next;

run->state='R';

}

void roundrun(char chose)

{

int i=1;

while(run!=NULL)

{

run->cputime+=1;

run->needtime-=1;

run->count+=1;

if(run->needtime==0)

{

run->next=finish;

finish=run;

run->state='F';

run->prio=0;

run=NULL;

if(ready!=NULL)

{

firstin();

}

}

else

{

if(run->count==run->round)

{

run->count=0;

if(ready!=NULL)

{

run->state='W';

insert\_rr(run);

firstin();

}

}

}

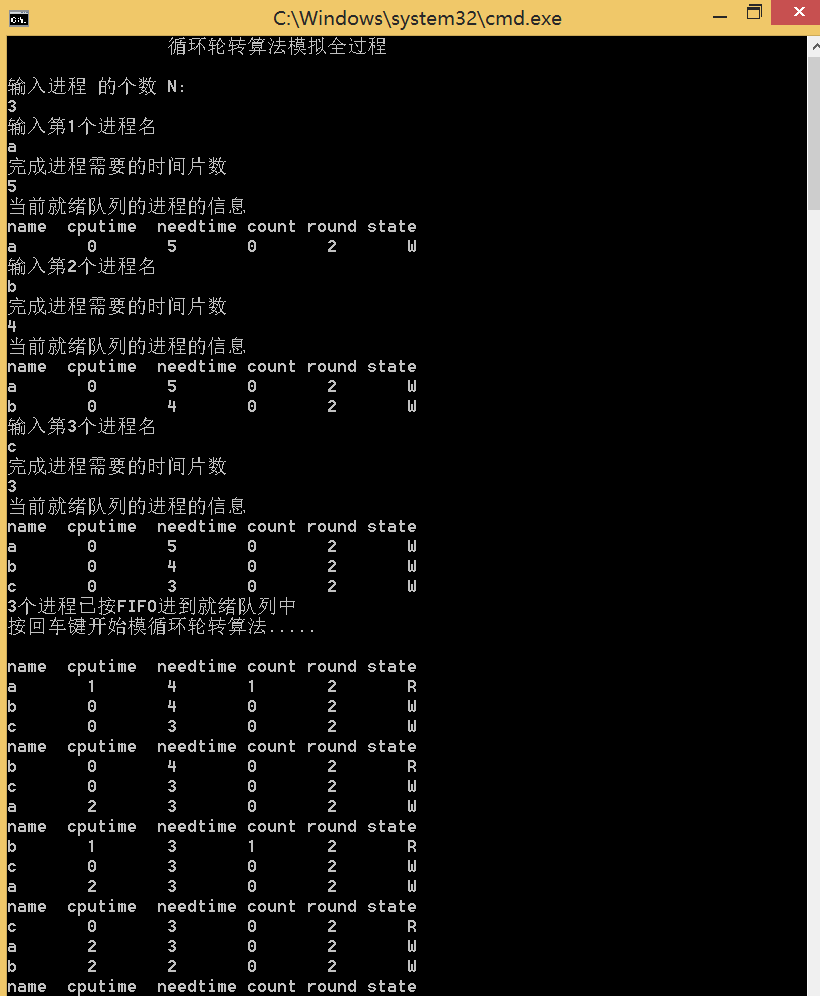
print(chose);

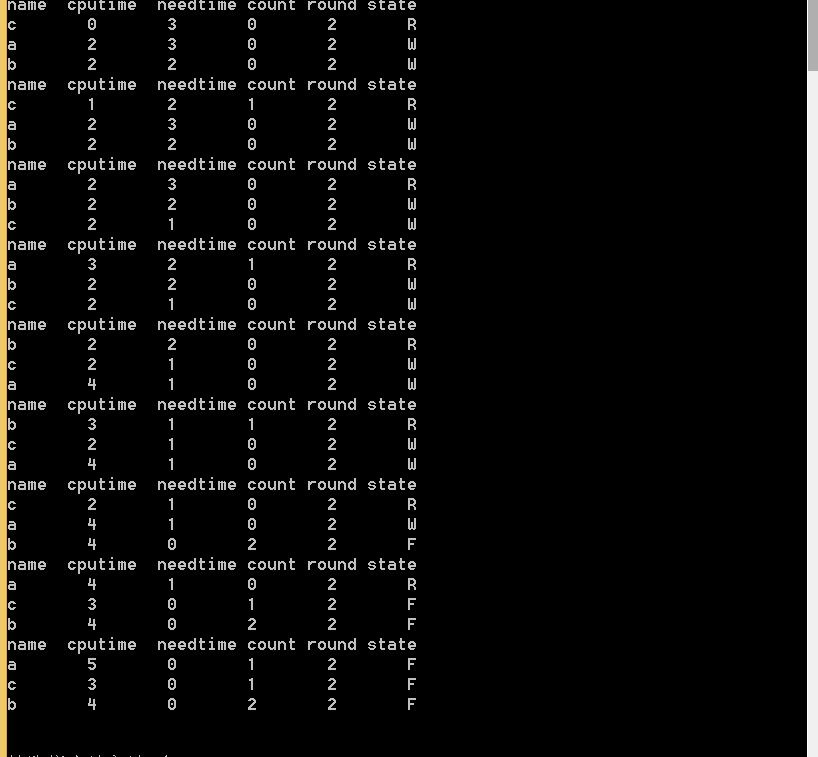
}

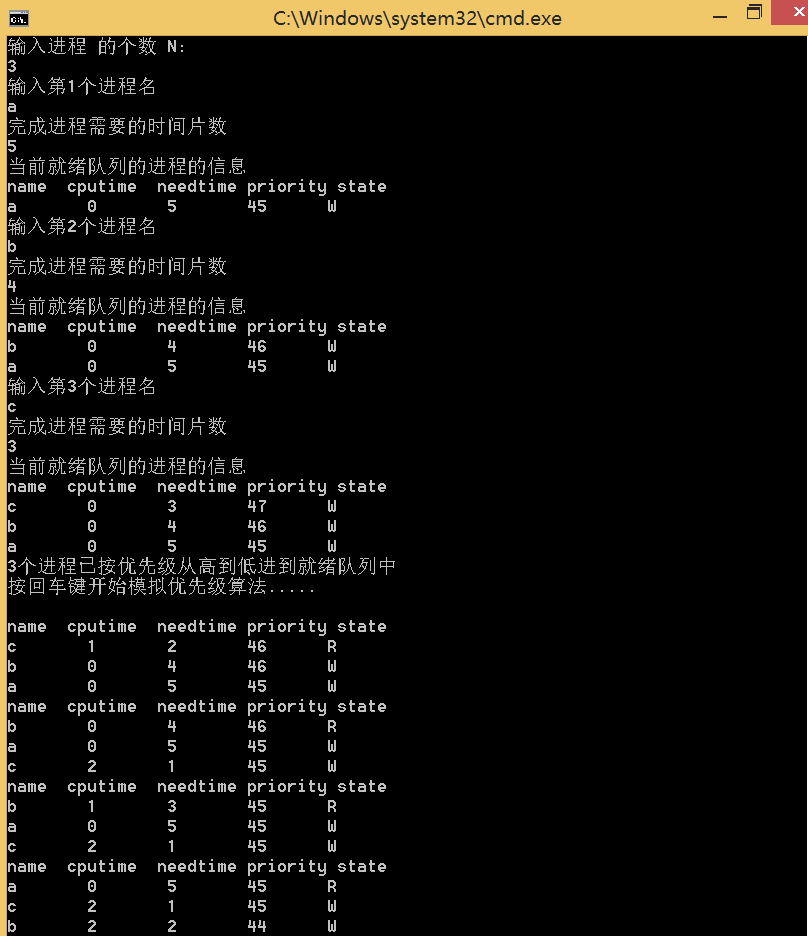
getchar();

}

四、实验结果









五、实验总结

通过这次实验，能够更好的理解了时间片轮转的原理，能够更好地认识如何通过优先数调度与循环轮转调度算法来进行进程调度，课上讲的知识，经过这次实验，加深了我的理解。能够把课上的知识应用到实际当中，还是很有收获的。刚开始做这个实验的时候，感觉无从下手，之后又仔细的阅读学习了两个算法，从网上找了一些相关的代码，最后整合了出来。